

CHAPTER 25

Management Resources

25.	CHAPTER 25	25-1
25.1	General	25-1
25.2	Structure of Management Resources	25-1
25.2.1	Public RFC-Based Management Resources	25-1
25.2.1.1	Public RFC Management Information Base (MIB) Support	25-1
25.2.1.2	Notifications Support	25-2
25.2.1.3	Table Management using the RowStatus Column	25-2
25.2.2	TmNS-Specific Management Resources	25-3
25.2.2.1	tmnsTmaCommon	25-3
25.2.2.2	tmnsTmaSpecificCapabilities	25-3
25.2.2.3	tmnsNetworkNode	25-4
25.2.2.4	tmnsGeneralNotification	25-5
25.3	Management Resource Matrix	25-5
25.3.1	Hierarchy Element Class	25-5
25.3.2	Resource Name	25-6
25.3.3	Parent Resource Name	25-6
25.3.4	Resource Position	25-6
25.3.5	Resource URN	25-6
25.3.6	MIB OID	25-7
25.3.7	Resource Syntax	25-7
25.3.8	Access Level	25-7
25.3.9	Default Value	25-8
25.3.10	Table Index #	25-8
25.3.11	Date Introduced	25-8
25.3.12	Persistent	25-8
25.3.13	Idempotency	25-8
25.3.14	Description	25-8
25.3.15	Comment	25-9
25.4	Management Protocols	25-10
25.4.1	SNMP-based <i>ManagementResources</i>	25-10
25.4.2	HTTP-based <i>ManagementResources</i>	25-10
25.4.3	TmNS Resource Management Protocols	25-13
25.4.3.1	Device Configuration Protocol	25-13
25.4.3.2	File Export Protocols	25-16
25.4.3.3	TmNS Configuration Negotiation Protocol	25-18
25.5	Uniform Resource Name (URN)	25-22

LIST OF FIGURES

Figure 25-1. TmNS-Specific Management Resources Hierarchy	25-3
Figure 25-2. SNMP-Based Management Resources Terminology Overview	25-10
Figure 25-3. HTTP-Based Management Resources Channel Overview	25-11
Figure 25-4. TmNS Configuration Negotiation Protocol Diagram	25-19

LIST OF TABLES

Table 25-1. RowStatus Values Overview	25-2
Table 25-2. Hierarchy Element Classes	25-5
Table 25-3. Management Resource Access Levels	25-7
Table 25-4. Required and Optional Media Types	25-12

Distribution Statement A

Approved for public release: distribution unlimited.

25. CHAPTER 25

Management Resources

25.1 General

Each *TmNS Manageable Application (TMA)* defines a set of Management Resources where each Management Resource defines application-specific data accessible via an application layer protocol. All TmNS-specific Management Resources reside within the *TmNS Management Resources Hierarchy*, which is defined in Appendix 25A. Additionally, TmNS components may be required to provide host management resource. In all cases, management resources are used to provide a uniform and interoperable method for managing components and aspects of the TmNS system. There are two primary protocols for accessing the management resources, SNMP and HTTP which uses a RESTful architecture.

The TmNS-specific management resources are maintained in the spreadsheet of Appendix 25A. The spreadsheet provides a simple interface for maintaining each of the individual management resources. Each row in the spreadsheet describes a different management resource. The spreadsheet can be used to generate an ASN.1-formatted text file that serves as the TMNS-MIB for SNMP application. The spreadsheet contains additional mapping information, such as URNs, for support of other management protocols.

25.2 Structure of Management Resources

The structure of management resources is hierarchical. The TmNS-specific management resources are defined in detail in this standard. Additional management resources are defined through references to pre-existing RFCs. As a matter of interoperability, the hierarchy of pre-existing RFCs is used in an unmodified fashion.

25.2.1 Public RFC-Based Management Resources

25.2.1.1 Public RFC Management Information Base (MIB) Support

Several management resources at the host level are defined in public RFC MIBs. *TMAs* that implement NetworkNode management capabilities shall provide the following host-level management resources:

- SNMPv2-MIB (RFC 3418) snmpBasicComplianceRev2
- IF-MIB (RFC 2863) ifCompliance3
- IP-MIB (RFC 4293) ipMIBCompliance2
- TCP-MIB (RFC 4022) tcpMIBCompliance2
- UDP-MIB (RFC 4113) udpMIBCompliance2

Related RFCs:

- RFC 3418: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)
- RFC 2863: The Interfaces Group MIB
- RFC 4293: Management Information Base for the Internet Protocol (IP)
- RFC 4022: Management Information Base for the Transmission Control Protocol (TCP)
- RFC 4113: Management Information Base for the User Datagram Protocol (UDP)

25.2.1.2 Notifications Support

All *TMA*s shall be capable of generating SNMP notifications. All *TMA*s shall implement the following MIB groups:

- SNMP-TARGET-MIB::snmpTargetBasicGroup
- SNMP-TARGET-MIB::snmpTargetResponseGroup
- SNMP-TARGET-MIB::snmpTargetCommandResponderGroup
- SNMP-TARGET-MIB::snmpTargetNotifyGroup
- SNMP-TARGET-MIB::snmpTargetNotifyFilterGroup

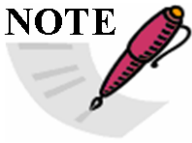
Related RFCs:

- RFC 3413: Simple Network Management Protocol (SNMP) Applications

25.2.1.3 Table Management using the RowStatus Column

*TMA*s that include tables with a RowStatus column shall implement the RowStatus column operation in accordance with the following:

- RFC 2579, Textual Conventions for SMIv2

 <p>NOTE</p>	<p>The RowStatus column is used to manage the creation and deletion of table rows as well as provide status. Table 25-1 provides an overview of the RowStatus values for quick reference. Refer to RFC 2579 for additional information.</p>			
	<p>Table 25-1. RowStatus Values Overview</p>			
	<p><i>Value</i></p>	<p><i>Description</i></p>	<p><i>Command</i></p>	<p><i>Status</i></p>
	<p>active</p>	<p>Row is accessible</p>	<p>✓</p>	<p>✓</p>
	<p>notInService</p>	<p>Row exists but is not currently accessible</p>	<p>✓</p>	<p>✓</p>
	<p>notReady</p>	<p>Row exists but is missing information</p>	<p>✗</p>	<p>✓</p>
	<p>createAndGo</p>	<p>Create a new row and have the row's status set to 'active'</p>	<p>✓</p>	<p>✗</p>
	<p>createAndWait</p>	<p>Create a new row and have the row's status set to 'notReady'</p>	<p>✓</p>	<p>✗</p>
	<p>destroy</p>	<p>Delete a row</p>	<p>✓</p>	<p>✗</p>

25.2.2 TmNS-Specific Management Resources

All management resources that are TmNS-specific fall under the top-level hierarchy element “tmns”. These resources are categorized into the four sub-categories presented in Figure 25-1.

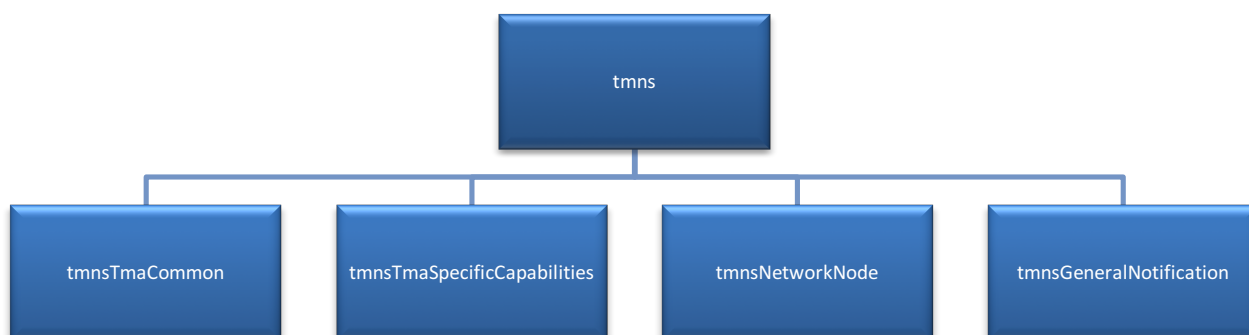
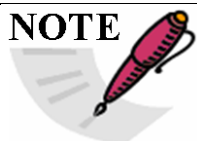


Figure 25-1. TmNS-Specific Management Resources Hierarchy

NOTE



Only the first level sub-containers of management resources are mentioned in the sections below. As a matter of consolidating documentation, considerably more detail is provided in the Management Resource Matrix.

25.2.2.1 tmnsTmaCommon

The tmnsTmaCommon resource is a container of management resources that shall be available on all *TMA*s unless otherwise noted. It contains the following six resource containers:

- tmnsTmaCommonIdentification
- tmnsTmaCommonFault
- tmnsTmaCommonConfiguration
- tmnsTmaCommonControl
- tmnsTmaCommonStatus
- tmnsTmaCommonSecurity

All TmNS-specific management resources contained within this resource container are found in the Management Resource Matrix of Appendix 25A.

25.2.2.2 tmnsTmaSpecificCapabilities

The tmnsTmaSpecificCapabilities resource is a container of management resources that define management resources for capabilities that are application-specific. Resource containers that reside under the tmnsTmaSpecificCapabilities resource group management resources by capabilities. These resource containers are:

- tmnsNetworkFabricDevice
- tmnsACU

- tmnsDAU
- tmnsRecorder
- tmnsMasterClock
- tmnsSSTTx
- tmnsSSTRx
- tmnsAdapter
- tmnsRCDataSource
- tmnsLTCDDataSource
- tmnsLTCDDataSink
- tmnsConsolidatedManager
- tmnsRadio
- tmnsLinkManager
- tmnsRCDataSink
- tmnsVoiceGateway
- tmnsRFNetworkManager
- tmnsTPA
- tmnsPCMGateway
- tmnsNetworkGateway
- tmnsRFNOManager
- tmnsRAN
- tmnsTmnsSourceSelector

A *TMA* that supports a resource container shall support all management resources within that resource container unless otherwise noted.

All TmNS-specific management resources contained within this resource container are found in the Management Resource Matrix of Appendix 25A.

25.2.2.3 **tmnsNetworkNode**

The tmnsNetworkNode resource is a container of management resources that provide status and control capabilities that are specific to the host machine. For *NetworkNodes* that only run a single *TMA*, the *TMA* shall implement all management resources contained within the tmnsNetworkNode resource container. If more than one *TMA* are executed concurrently on a single *NetworkNode*, only one *TMA* is required to implement the management resources contained within the tmnsNetworkNode resource container. *TMAs* that implement the tmnsNetworkNode resource container shall support all management resources within the tmnsNetworkNode resource container unless otherwise noted. The four resource containers contained within tmnsNetworkNode are the following:

- tmnsNetworkNodeIdentification
- tmnsNetworkNodeConfiguration
- tmnsNetworkNodeControl
- tmnsNetworkNodeStatus

All TmNS-specific management resources contained within this resource container are found in the Management Resource Matrix of Appendix 25A.

25.2.2.4 **tmnsGeneralNotification**

All *TMA*s shall be capable of generating event-based notifications. Management resources regarding general notifications are contained within the tmnsGeneralNotifications container resource. This container resource contains the following nine resource containers:

- configurationCompleteNotificationBranch
- timeLockLostNotificationBranch
- ieee1588MaxOffsetFromMasterNotificationBranch
- ieee1588MaxJitterNotificationBranch
- tempOutOfRangeNotificationBranch
- accessAnomalyDetectionNotificationBranch
- powerFaultNotificationBranch
- invalidInputNotificationBranch
- configurationChangeNotificationBranch

All TmNS-specific management resources contained within this resource container are found in the Management Resource Matrix of Appendix 25A.

25.3 **Management Resource Matrix**

The management resource matrix is the table that defines all TmNS-specific management resources. Each row in the matrix represents a management resource. Each column describes the resource. The matrix can be used to auto-generate the ASN.1-formatted TMNS-MIB file which shall be used by applications that use the SNMP protocol. The columns are described in more detail in this section.

25.3.1 **Hierarchy Element Class**

This field indicates the class of the management resource with respect to its structure in the management resource hierarchy. The possible values are provided in Table 25-2.

Table 25-2. Hierarchy Element Classes

Value	Name	Description
B	Branch	A branch in the management resource hierarchy that may contain child entries
I	Identity	
S	Scalar	A leaf node in the management resource hierarchy.
N	Notification	A management resource that is used for asynchronous reporting of management resources based on some triggering condition.
T	Table	A hierarchical structure of management resources that may be duplicated across several instances. Management resources that comprise a table are the table sub-elements, each of which comprise a column of the table. Rows of a table correspond to each


		distinct instance of the collection of table sub-element management resources. Rows are unique based on a unique combination of the table's defined index values. A table may contain more than one index value in order to guarantee row uniqueness.
ts	Table Sub-element	An element, of scalar type, that comprises a column of the parent table
TC	Textual Convention	A syntax definition that associates specific constraints with its type. Often these constraints resolve to an integer enumeration. The textual convention may be used as a valid resource syntax for other management resources.

25.3.2 Resource Name

This field contains the name of the management resource. The management resource name shall be unique across all TmNS-specific management resources.

The resource name shall map to the name of the MIB variable within the TMNS-MIB. Similarly, management resource names of the public RFC MIBs are already known.

HTTP-based names beginning with “tmns” shall be considered as a short-cut to the longer equivalent name enforced by the TMNS-MIB. That is, iso:org:dod:internet:private:enterprises:tmns.

 NOTE	Resource names in the Management Resource Matrix have been chosen such that they are compatible with both known targets: SNMP and HTTP. SNMP MIBs require uniqueness for all names within a MIB. The intention is for the management resource names to match that of the MIB variable names.
---	--

25.3.3 Parent Resource Name

This field shall contain the name of its parent resource within the management resource hierarchy.

25.3.4 Resource Position

This field represents the resource's child position with respect to its parent resource. The value of this field shall be an integer greater than zero and are not required to be sequential. The resource position shall be unique amongst all resources that share a common parent resource.

25.3.5 Resource URN

This field contains the URN associated with the resource. The syntax for TmNS-specific management resources is defined in Section 25.5.

25.3.6 MIB OID

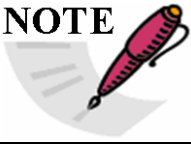
This field represents the numerical hierarchy associated with the resource, beginning with the numerical value associated with the root of the TmNS-specific management resource tree, “tmns” which has a value of 31409. For the complete MIB OID, see Section 25.4.1.

25.3.7 Resource Syntax

This field represents the syntax associated with the resource. Resources may utilize a syntax with constraints as well as syntax types that are defined by textual conventions within a supported public RFC or within the TmNS. Example of syntax constraints may be in size limitation, range of acceptable values, and enumerations.

Resources that are textual conventions defined by the TmNS are not accessible resources for reading or writing. As such, these resources do not exist in the hierarchy of managed resources, e.g. they have neither a parent resource association nor a resource position.

NOTE



Resource Syntax in the Management Resource Matrix have been chosen such that they reflect the syntax type constraints associated with the MIB definition of the resources.

25.3.8 Access Level

This field contains the type of access associated with the resource. The list of possible access levels and their descriptions are provided in Table 25-3.

Table 25-3. Management Resource Access Levels

Value	Description
read-only	The resource only supports reading and cannot be written.
read-write	The resource supports both reading and writing.
read-create	The resource supports both reading and writing and resides within a table that allows the creation of new rows (instances) through management via application layer protocols.
not-accessible	The resource does not support reading or writing. These resources are typically associated with tables and do have an associated syntax for the purpose of hierarchy structure.
<blank>	Resources that define textual conventions or only provide structure, such as parent resources, shall be left blank.

25.3.9 Default Value

Default values are given for all readable resources unless otherwise indicated. For instance, the default value for a table is an “empty” state because it has no rows.

In the case of read-only resources that report status, the defaults shall be applied during *TMA* initialization; the actual status value shall replace the default value once the *TMA* is able to acquire that status.

In the case of configuration and readable control resources, the default values listed shall be applied to the *TMA* when a *TMA* “Reset to Default” is executed.

25.3.10 Table Index #

This field shall be used by any table sub-element that serves as an index into the table. The value shall be an integer that indicates its index position in relation to any other indexes associated with the table.

For any resource that does not serve as an index into a table, this value shall be left blank.

25.3.11 Date Introduced

This field identifies the version of the standard in which the particular management resource was introduced into the standard. This is intended to aid in interoperability as the standard is updated and new resources are added or existing resources are updated.

25.3.12 Persistent

If the *Persistent* property is true, the resource’s value shall be retained across resets (including host loss of power) except when a *TMA* “Reset to Default” is executed. *TmNS* Management Resources shall not be persistent except where specifically designated. Resources designated as persistent shall have their value stored in non-volatile memory whenever the resource is written. Persistent resources shall not retain their value when a *TMA* “Reset to Default” is executed.

25.3.13 Idempotency

A resource with the *Idempotency* property set to “true” indicates that a readable resource can be read multiple times without affecting the resource’s value and that a writeable resource can be written multiple times without adverse consequences. The *Idempotency* property shall apply to all *TmNS*-specific Management Resources except where specifically noted.

25.3.14 Description

This field describes the management resource. For some resources, specific behaviors and/or relationships to other management resources are defined. This field shall be used for documentation of the management resource. A description shall be provided for each management resource.

25.3.15 Comment

This field provides additional comments that may accompany a management resource or group of resources. Comments shall not include information that is needed for understanding how to use a particular resource or set of resources.

25.4 Management Protocols

Two application layer protocols provide access to the *ManagementResources*: SNMP and HTTP.

25.4.1 SNMP-based *ManagementResources*

*TMA*s that provide or access SNMP-based management resources shall comply with the SNMP requirements specified in Chapter 22, Network-Based Protocol Suite. The TmNS Management Information Base (TMNS-MIB) contains all TmNS-specific management resources. At the top of the TmNS-specific management resource hierarchy is the resource “tmns”.

The TMNS-MIB has the following Object Identifier (OID) registered with the Internet Assigned Number Authority (IANA):

Telemetry Network System (tmns): iso.org.dod.internet.private.enterprise.31409 (1.3.6.1.4.1.31409)

Documentation for the TMNS-MIB is part of the Management Resource Matrix found in Appendix 25A. An ASN.1 formatted file can be generated from the Management Resource Matrix and shall contain the available documentation for each resource identified by the TMNS-MIB.

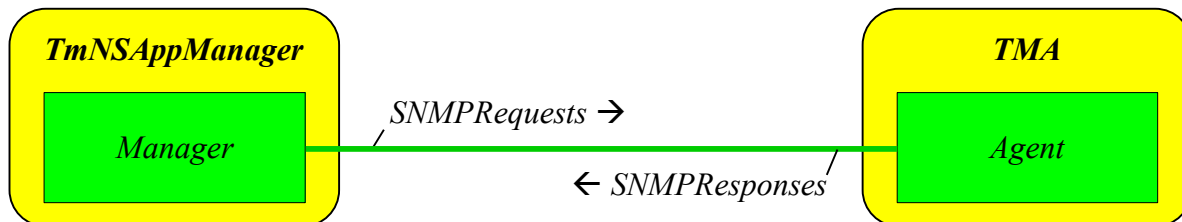


Figure 25-2. SNMP-Based Management Resources Terminology Overview

25.4.2 HTTP-based *ManagementResources*

*TMA*s that provide or access HTTP-based resources shall comply with the HTTP requirements specified in Chapter 22, Network-Based Protocol Suite.

A *ResourceChannel* identifies a network connection used to transport *ResourceRequests* and *ResourceResponses* between a *ResourceClient* and a *ResourceServer*. *ResourceClients* and *ResourceServers* using the *ResourceChannel* shall exchange *ResourceRequests* and *ResourceResponses* using the Hypertext Transport Protocol (HTTP), as specified in Chapter 22, Network-Based Protocol Suite.

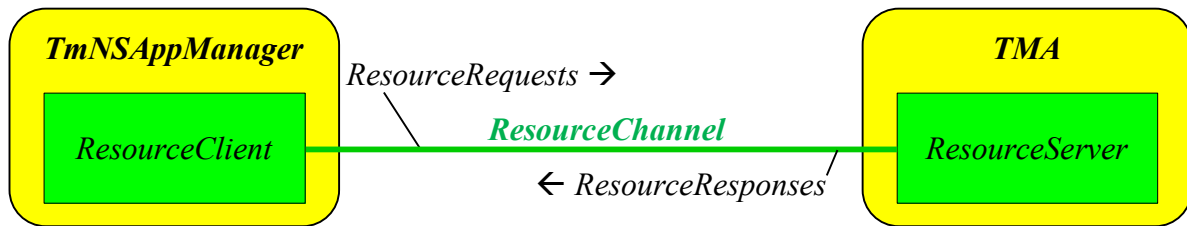


Figure 25-3. HTTP-Based Management Resources Channel Overview

The *ResourceClient* shall act as the HTTP client and the *ResourceServer* shall act as the HTTP server. Each *TMA* shall include a *ResourceServer*.

ResourceClients and *ResourceServers* shall transport *ResourceRequests* and *ResourceResponses* in the *ResourceChannel* using TCP.

The *ResourceChannel* shall use the same Differential Service Code Points (DSCPs) in both directions based on the DSCP selected by the *ResourceClient*.

The *ResourceChannel* shall support the following HTTP methods: GET, PUT, POST, and DELETE. Support for other HTTP methods is not required. The HTTP methods used in the *ResourceRequest* shall use the *TmNS_Request_Defined_URI* to access *ResourceServer* resources.

Key *ResourceRequest* HTTP Request Headers:

Request Header	Value	Comments
Host	Domain name and TCP port of <i>ResourceServer</i> .	Required for all HTTP/1.1 requests
Accept	Media Type(s) (i.e., Content-Types) acceptable in the <i>ResourceResponse</i> .	See Media Type discussion below

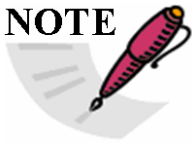
If a *ResourceRequest* or *ResourceResponse* includes an Entity Body, the following HTTP Headers shall be in the *ResourceRequest* or *ResourceResponse* respectively:

Response Header	Value	Comments
Content-Type	The Media Type of the <i>ResourceResponse</i> body.	See Media Type discussion below
Content-Length	Length of <i>ResourceResponse</i> body in bytes.	
Location	Used in redirection	Primarily used for resource creation and asynchronous operations

Table 25-4. Required and Optional Media Types

Required Media Types		
Media Type	Comments	Common Abbr
application/vnd.tmns.mdl+xml	IANA-registered Media Type for <i>TmNS Metadata Language</i>	MDL
application/vnd.tmns.arl+xml	IANA-registered Media Type for <i>TmNS Management Resources Language</i>	ARL*
Optional Media Types		
Media Type	Comments	
application/vnd.tmns.ihal+xml	IANA-registered Media Type for TmNS Instrumentation Hardware Abstraction Language	IHAL
application/xml	Generic XML document exchange	
text/html	Serve HTML pages to a web browser	
text/plain	Web browser support via Javascript or similar protocol	
others	Other Media Types may be implemented at the vendor's discretion (although other representations are outside the scope of these standards)	

* Documented in Appendix CC, Management Resources Language (ARL) XML Schema.

 NOTE	Supporting multiple Accept header values provides a <i>ResourceServer</i> the capability to support multiple interfaces for the same resource. For example: the GET {rootPath}/dataChannel method could return a Media Type of “text/html” and thereby provide the Data Channel List as an HTML page (i.e., web page) rather than as an XML document.
---	---

If a *ResourceServer* receives a *ResourceRequest* for an unrecognized or unsupported *Resource*, the *ResourceServer* shall return a Status Code of 404, Not Found.

If a *ResourceServer* receives a *ResourceRequest* with an unrecognized URI Parameter (*TmNSparam*), the *ResourceServer* shall return an Error Response with all pertinent information included in the Error Message and a Status Code of 400, Bad Request.

If a *ResourceServer* receives a *ResourceRequest* and is unable to process the request due to an internal *ResourceServer* problem, the *ResourceServer* shall return an Error Response with all pertinent information included in the Error Message and a Status Code of 500, Internal Server Error.

25.4.3 TmNS Resource Management Protocols

25.4.3.1 Device Configuration Protocol

*TMA*s shall support the transfer of configuration files (e.g., *MDL Instance Documents*) using the File Transfer Protocol (FTP) as specified in Chapter 22, Network-Based Protocol Suite.

*TMA*s should support the transfer of configuration files using the Hypertext Transport Protocol (HTTP) as specified in Chapter 22, Network-Based Protocol Suite.

25.4.3.1.1 Configuration Protocol for *TMA*s

The *TMA* Configuration Protocol is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to configure the target *TMA* using an *MDL Instance Document*.

The *TMA* Configuration Protocol is comprised of the following steps:

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationURI** resource on the target *TMA* to the location of the configuration file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource on the target *TMA* to “true”. Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to “true”, any attempt by the *TmNSApp* manager to change the resource’s value shall be ignored until the target *TMA* has set the resource’s value to “false”.

NOTE



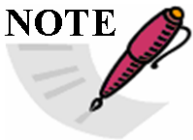
To cancel the Configuration Process, a *TmNSApp* manager may execute either a *TMA* Reset or a *TmNSHost* Reset.

3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource being set to true, the *TMA* shall retrieve the configuration file indicated by the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationURI** resource. If a retrieval error occurs, the *TMA* shall follow the steps outlined in Section 25.4.3.1.1.1.
4. Upon successful retrieval of the configuration file, the *TMA* parses and checks the retrieved configuration file. The *TMA* is not required to perform an XML validation of the configuration file (the *TMA* may assume the configuration is valid with respect to its schema). If an anomaly is detected, the *TMA* shall follow the steps outlined in Section 25.4.3.1.1.1.

5. The *TMA* applies the changes found in the configuration file. If an error is detected, the *TMA* shall follow the steps outlined in Section 25.4.3.1.1.1.
6. When all changes have been successfully applied to the *TMA* (i.e., configuration is complete), the *TMA* shall:
 - a. Update the *TMA*'s **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion** resource according to the format specified in the description of the this resource in the Management Resource Matrix.
 - b. Set the *TMA*'s **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber** resource to "2" and **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resource to "Configured".
 - c. Set the *TMA*'s **tmnsTmaCommon:tmnsTmaCommonConfiguration:configChangeCounter** resource to "0".
 - d. Set the *TMA*'s **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to "false".
 - e. Send a configurationCompleteNotification via the **tmnsGeneralNotification:configurationCompleteNotificationBranch:configurationCompleteNotifications:configurationCompleteNotification** resource. The notification shall indicate a successful configuration attempt.

If a configuration error occurs, the *TMA* shall follow the steps outlined in Section 25.4.3.1.1.1.

NOTE



A *TMA* is only required to store configuration information applicable to itself (i.e., storing configuration information of other TMAs is not required).

25.4.3.1.1.1 Configuration Error Handling

If the *TMA* detects an error during the Configuration Process, the *TMA* shall adhere to the following steps:

1. The *TMA* shall follow one of the two following approaches in this step:
 - a. The *TMA* shall attempt to restore the previous configuration prior to the initiating of the configure attempt. If the *TMA* is able to restore the previous configuration, the *TMA* shall set its

tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion, **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber**, and **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resources to their previous values prior to the initiation of the Configuration Process. If the *TMA* was actively publishing or subscribing to *TmNSDataMessages* prior to the initiating of the configuration attempt, it shall not return to that mode of operation. Rather, a *TMA* that recovers from a failed configuration attempt shall not begin publishing or subscribing to *TmNSDataMessages* until further commanded to do so by a *TmNSApp* manager. If the *TMA* is unable to restore the previous configuration as described, the *TMA* shall utilize the other error handling approach.

- b. The *TMA* shall set its

tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationVersion resource to an empty string in accordance with the description of the resource in the Management Resource Matrix. The *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateNumber** resource to “1” and its **tmnsTmaCommon:tmnsTmaCommonStatus:tmaStateString** resource “Unconfigured”. The *TMA* shall not publish or subscribe to any *TmNSDataMessages* until after a successful configuration attempt.

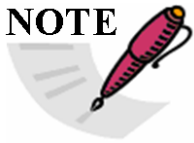
NOTE



A *TMA* is not required to restore any previous state after a configuration failure. Approach 1a is expected to be used by *TMA*s that are capable of restoring the previous configuration state.

2. The *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultNumber** and **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString** resources to the appropriate value into a row in the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable**.
3. The *TMA* shall set its **tmnsTmaCommon:tmnsTmaCommonConfiguration:configure** resource to “false”.
4. The *TMA* shall send a *configurationCompleteNotification* via the **tmnsGeneralNotification:configurationCompleteNotificationBranch:configurationCompleteNotifications:configurationCompleteNotification** resource. The notification shall indicate a failed configuration attempt.

NOTE



The following are examples of possible configuration errors

- a. The transfer of the configuration file fails.
- b. An incomplete or invalid configuration file is received.

	c. A value specified in the configuration file conflicts with a <i>TMA</i> constant or allowable value range.
--	---

25.4.3.2 File Export Protocols

*TMA*s shall support the exporting of files via the processes defined in the following subsection:

- Export Configuration File Protocol for *TMA*s
- Export Log File Protocol for *TMA*s

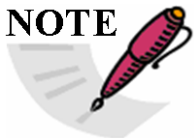
25.4.3.2.1 Export Configuration File Protocol for *TMA*s

The Export Configuration File Protocol for *TMA*s is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to retrieve the target *TMA*'s configuration via an *MDL Instance Document*.

The Export Configuration File Protocol is comprised of the following steps:

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationExportURI** resource on the target *TMA* to a destination location for the configuration file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource on the target *TMA* to "true". Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource to "true", any attempt by the *TMA* manager to change the resource's value shall be ignored until the target *TMA* has set the resource's value to "false".


NOTE



To cancel the Export Configuration File Process, a *TmNSApp* manager may execute either a *TMA* Reset or a *TmNSHost* Reset.

3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource being set to "true", the *TMA* shall send an MDL file that contains the description of the *TMA*'s current configuration to the destination location indicated by the **tmnsTmaCommon:tmnsTmaCommonConfiguration:configurationExportURI** resource.
4. Upon completion of the file transfer process (successful or failed), the *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonConfiguration:exportConfiguration** resource to "false".

5. If an error occurs, the *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultNumber** and **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString** resources to the appropriate value into a row in the **tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable**.

 NOTE	<p>The full state of the <i>TMA</i> is represented by its stored configuration information (i.e., information transportable via an <i>MDL Instance Document</i>) and the state of the <i>TMA</i>'s resources. Thus, it may be necessary for a <i>TmNSApp</i> manager to retrieve the current values of a <i>TMA</i>'s resources in conjunction with retrieving its configuration file via the export process.</p>
---	---

25.4.3.2.2 Export Log File Protocol for *TMA*s

The Export Log File Protocol for *TMA*s is a sequence of steps executed between a *TmNSApp* manager and a target *TMA* to retrieve the target *TMA*'s log file.

The Export Log File Protocol is comprised of the following steps:

1. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonControl:logFileExportURI** resource on the target *TMA* to a destination location for the log file.
2. The *TmNSApp* manager sets the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource on the target *TMA* to "true". Once a *TmNSApp* manager has set the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource to "true", any attempt by the *TmNSApp* manager to change the resource's value shall be ignored until the target *TMA* has set the resource's value to "false".

 NOTE	<p>To cancel the Export Log File Process, a <i>TmNSApp</i> manager may execute either a <i>TMA</i> Reset or a <i>TmNSHost</i> Reset.</p>
---	--

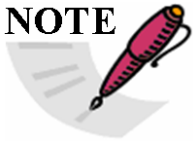
3. Upon receipt of the **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource being set to "true", the *TMA* shall send its log file to the destination location indicated by the **tmnsTmaCommon:tmnsTmaCommonControl:logFileExportURI** resource.
4. Upon completion of the file transfer process (successful or failed), the *TMA* shall set the *TMA* **tmnsTmaCommon:tmnsTmaCommonControl:exportLogFile** resource to "false".
5. If an error occurs, the *TMA* shall set the **tmnsTmaCommon:tmnsTmaCommonFaultactiveFaultsTable:faultNumber** and

tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable:faultString
resources to the appropriate value into a row in the
tmnsTmaCommon:tmnsTmaCommonFault:activeFaultsTable.

25.4.3.3 TmNS Configuration Negotiation Protocol

NetworkNodes that sample and package data and *TmNSAppManagers* that construct MDL files shall implement the TmNS Configuration Negotiation Protocol. The protocol consists of a dialog between the *TmNSAppManager* and the data acquisition *NetworkNode*. The protocol is used to communicate the desired set of measurements to be produced and the capability of the acquisition device to provide the data at the requested rates.

NOTE



The data acquisition *NetworkNode* may be represented by a physical component or software emulation.

The communication between the negotiating entities utilizes HTTP (Chapter 22.5.2.2), SNMP (Chapter 22.5.2.1), and FTP (Chapter 22.5.2.4). The communication workflow is depicted in Figure 25-4.

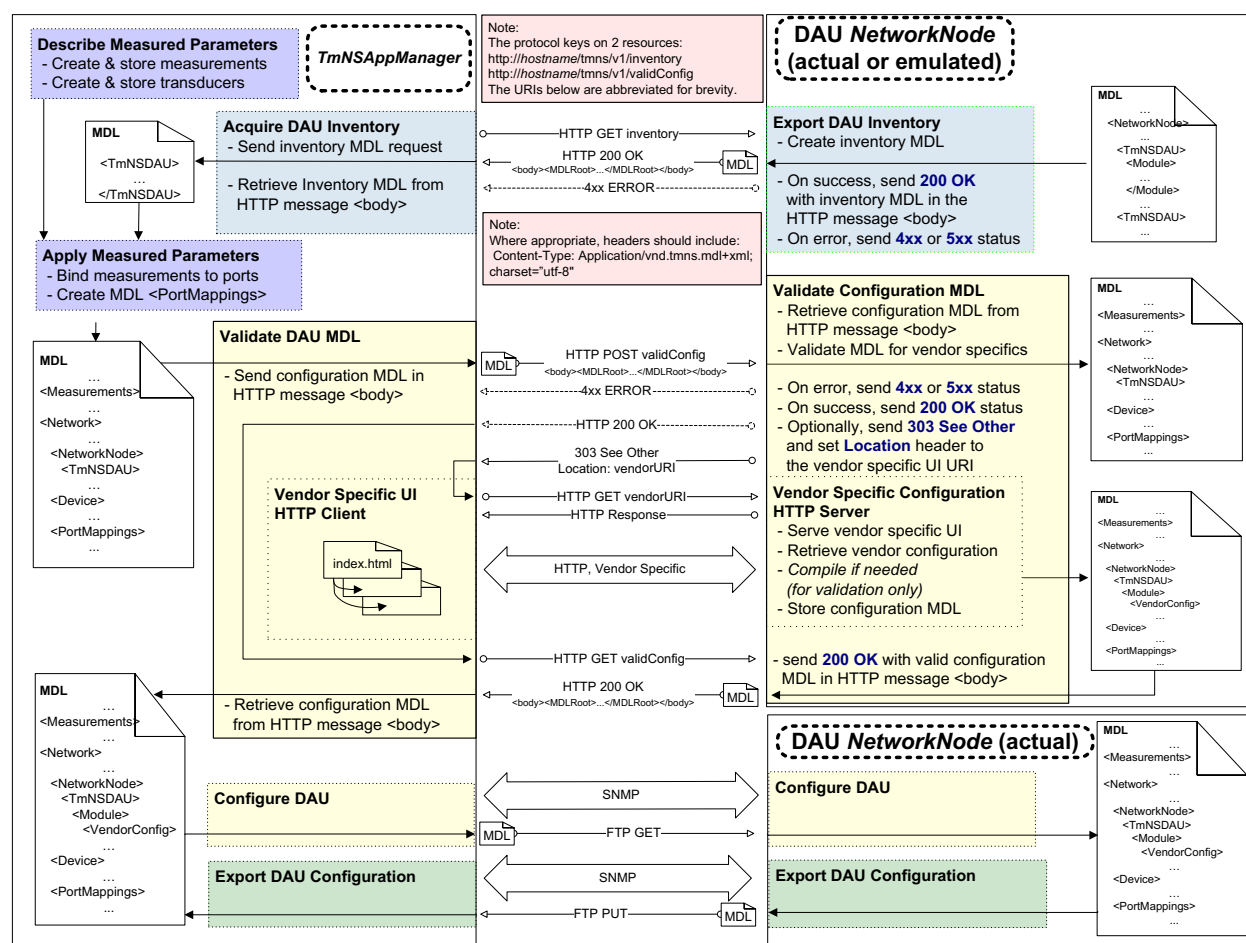


Figure 25-4. TmNS Configuration Negotiation Protocol Diagram

The *TmNS* Configuration Negotiation Protocol is a sequence of steps executed between a *TmNSAppManager* and a data acquisition *NetworkNode* to build a valid MDL instance document containing the data acquisition *NetworkNode* configuration.

The *TmNS* Configuration Negotiation Protocol is comprised of the following steps:

1. The *TmNSAppManager* retrieves inventory from the data acquisition *NetworkNode* by accessing the Inventory Resource on data acquisition *NetworkNode*.
2. The *TmNSAppManager* binds measurement information to the data acquisition *NetworkNode* inventory, creating a candidate for the data acquisition *NetworkNode* configuration.
3. The *TmNSAppManager* sends the candidate configuration to the Valid Configuration Resource on the data acquisition *NetworkNode*.
4. If the candidate configuration is not considered valid by the data acquisition *NetworkNode*, it may return an error or it may redirect the *TmNSAppManager* to a vendor-specific URI. The interaction between the *TmNSAppManager* and the

vendor-specific URI is outside the scope of this standard.

5. If the candidate configuration is considered valid by the data acquisition *NetworkNode*, it will indicate success to the *TmNSAppManager* and store the valid configuration without actually configuring.
6. The *TmNSAppManager* retrieves the valid configuration from the Valid Configuration Resource on the data acquisition *NetworkNode*.
7. The *TmNSAppManager* may configure the data acquisition *NetworkNode* with the valid configuration via the *TMA* Configuration Protocol (see *reference*).

25.4.3.3.1 TmNS Inventory

Data acquisition *NetworkNodes* shall document inventory in an MDL instance document by implementing the Inventory Resource at the URI, */tmns/v1/inventory*. The Inventory Resource shall support the HTTP GET method. The Inventory Resource shall indicate success by returning a *200 OK* response containing the inventory MDL instance document in the body. The data acquisition *NetworkNode* may indicate errors by returning an appropriate *4xx* or *5xx* status code response.

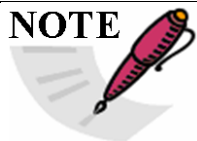
25.4.3.3.2 TmNS Configuration Validation

Data acquisition *NetworkNodes* shall augment the *TmNS* Configuration Protocol (see *reference*) by implementing the Valid Configuration Resource at the URI, */tmns/v1/validConfig*. The Valid Configuration Resource shall support the HTTP POST and GET methods. This resource shall validate the configuration in an MDL instance document when accessed by a POST method. This resource shall return the most recently validated MDL instance document when accessed by a GET method.

The Valid Configuration Resource shall validate the content sent within the body of a POST request. The Valid Configuration Resource may indicate success by returning a *200 OK* response. The Valid Configuration Resource may indicate errors by returning an appropriate *4xx* or *5xx* status code response. The Valid Configuration Resource may indicate detailed warnings and/or errors within the body of the response. The Valid Configuration Resource may provide a Vendor Specific interface. To indicate use of a Vendor Specific interface, the Valid Configuration Resource shall return a *303 See Other* response with the *Location* field set to the URI of the Vendor Specific interface.

The Valid Configuration Resource shall indicate success to a GET request by returning the most recently validated MDL instance document within the body of a *200 OK* response. The Valid Configuration Resource shall indicate that no valid MDL instance document exists by returning a *428 Precondition Required* response. The Valid Configuration Resource may indicate errors by returning an appropriate *4xx* or *5xx* status code response. The Valid Configuration Resource may indicate an alternate URI for the most recently validated MDL instance document by returning a *303 See Other* response with the *Location* field set to the alternate URI.

NOTE



Given that the validConfig resource processes data and can modify content within the Vendor Specific Interface, the constraints on PUT are too restrictive for the TmNS Configuration Protocol. Thus, the POST method was chosen over PUT.

25.5 Uniform Resource Name (URN)

The *TmNS* Management Resources Hierarchy uses the Uniform Resource Name (URN) defined in RFC 2141. The general syntax is specified below:

URN = "urn:" Namespace ID ":" Namespace Specific String (NSS)

For TmNS-specific management resources, the *TmNSURN*, "tmns" is assigned as the Namespace ID resulting in:

TmNSURN = "urn:tmns:" Namespace Specific String (NSS)

The Namespace Specific String (NSS) identifies a specific resource or set of resources under the *TmNS* Namespace. Examples:

- **urn:tmns:tmnsTmaCommon:tmnsTmaCommonIdentification** identifies all of the resources under the tmnsTmaCommonIdentification resource.
- **urn:tmns:tmnsTmaCommon:tmnsTmaCommonIdentification:tmaProductName** specifically identifies the tmaProductName resource.

To reduce documentation clutter, the "urn:tmns" is typically left off a resource's name. For example: the tmaProductName resource would be identified as the **tmnsTmaCommon:tmnsTmaCommonIdentification:tmaProductName** resource.